UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/851,405	05/08/2001	Young-seok Lee	YPLEE4.001AUS	1879

20995 7590 05/25/2004

KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614

EXAMINER	
MOIDUDDIN, NOREEN	
ART UNIT	**PAPER NUMBER**
2124	

DATE MAILED: 05/25/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/851,405	LEE, YOUNG-SEOK	
	Examiner	Art Unit	
	Noreen Moiduddin	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 May 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-9* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-9* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *08 May 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *4*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED OFFICE ACTION

1. Claims 1-9 have been examined.

Priority

2. Acknowledgment is made of applicant's claim for foreign priority under 35 U.S.C. 119(a)-(d). Application examined under priority date May 9, 2000.

Drawings

3. Figures 1-2 should be designated by a legend such as --Prior Art-- because only that which is old is illustrated. See MPEP § 608.02(g). A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Specification

4. The abstract of the disclosure is objected to because of minor misspellings.

   a. In line 1 applicant misspelled "therefore" as "therefor." "Therefor" is found in other parts of the application as well. Assume applicant means "therefore." Correction is required. See MPEP § 608.01(b).

5. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Claim Objections

6. Claims 1, 4, and 6 objected to because of the following informalities: minor misspelling. Claim 1 line 3, claim 4 line 3, and claim 6 line 3, applicant misspelled "functionalizes" as "functionizes." Assume applicant means "functionalizes". Appropriate correction is required.

Claim Rejections - 35 USC § 101

7. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1, 4, and 6 are rejected under 35 U.S.C. 101 because the claimed inventions are directed to non-tangible and non-statutory subject matter. The test systems disclosed in claims 1, 4, and 6 do not specify where the systems reside. Is the function library on a network, the object file on a sound wave, and the execution program on a computer? Where does each component of the system reside in respect to the target software running on a computer?

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) and the Intellectual Property and High Technology Technical Amendments Act of 2002 do not apply when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. Therefore, the prior art date of the reference is determined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

9. To expedite a complete examination of the application claim 1 rejected under 35 U.S.C. 101 (nonstatutory) above is further rejected as set forth below in anticipation of applicant amending this claim to place it within the four statutory categories of invention.

Claims 1-3 are rejected under 35 U.S.C. 102(e) as being anticipated by Rodrigues et al (U.S. 6,067,639).

As per claim 1, Rodrigues discloses a system:

a. **A function library file that functionizes and stores commands for executing objects of the target software as functions** (abstract, column 9 lines 20-28, lines 60-67, column 10, lines 1-4).

The test operation objects (executing objects) are instantiated by calls to functions in a test operation runtime library (function library file) (DLL). The test operation objects (executing objects) are derived from an object class provided in a programming library (test operation runtime library) of the present invention corresponding to functions to be tested within the computer application program (target software). "Functions are provided in the test operation object library (column 15, lines 56-57)." "Each group (test operation object groups) becomes associated with a unique group identifier (keyword)" (column 15, lines 60-61).

b. **An object file that sequentially records keywords, each of which indicates an object of the target software, in an order in which it is desired to test the target software, each of the keywords distinguished by an object identifier.** In playback auto-test mode or manual mode, a playback file (object file) is used or a new file (object file) created by a user through a user interface. "The playback file (object file) contains information (keywords distinguished by an object identifier) used by the methods of the present invention to reproduce a previously recorded (order in which it is desired to test the target software) test sequence" (column 13, lines 14-16). "Each entry in the playback file (object file) represents the invocation of a selected test operation object" (column 13, lines 27-29). When each entry represents the invocation of a selected test operation object, the word "represents" refers to an identifier that symbolizes "a selected test operation object." The identifier is a "keyword" that "represents" "a selected

test operation object". "Each group (test operation object groups) becomes associated with a unique group identifier (keyword)" (column 15, lines 60-61).

c. **An execution program that sequentially reads keywords from the object file, recognizes an object to execute, calls a function for executing the recognized object from the function library file, and executes the function** (column 12 lines 49-52, figure 7).

"The execute function (execution program) is defined by the product development group to perform whatever functional processing is appropriate to test the desired function corresponding to the test operation object (object from the function library file)." The terms reading and recognizing are inherently incorporated in the "functional processing" of the execute function (execution program). In order to read an "invocation of a selected test operation object (keyword)" and execute a function "corresponding to the test operation object", recognizing the selected test operation object is inherently incorporated. Without recognizing the object, the object cannot be executed. Figure 7 shows the "sequential process" of test operations (keywords) being executed from the "playback file" (object file). Item 700 prompts a user to pick a playback file (object file). Item 704 checks if "more test operations (keywords) [are] in Playback File (object file)". When the checking process occurs, the playback file (object file) is "read". If a test operation (keyword) is found, then item 706 executes the next test operation (keyword) listed. In order to execute, the execute function (execution program) must first recognize the test operation

object the test operation (keyword) refers to. The result is logged in a log file and then the process loops back to the beginning of item 704. Also see figure 6B, item 618. By definition playback files (object files) contain a sequentially prerecorded ordering of test operations (keywords).

As per claim 2, Rodrigues discloses "a computer operable method (software test method) for integrating and automating test procedures within a computer application program" (abstract, lines 1-2).

a. **Generating an object file wherein keywords are sequentially recoded in an order in which it is desired to test the target software, each keyword indicating an object of the target software and being distinguished by a respective object identifier** (figure 6B item 616, figure 7 items 704, 706, column 15, lines 27-30, column 20, lines 32-34). Figure 6B item 616 shows a generated test sequentially recording test operations (keywords) in a Playback file (object file). "Element 616 is next operable to record the selected test operation object (keyword) in a playback file (object file). The prerecorded ordering is maintained to reproduce a result or test a change to a program. Each entry in the playback file (object file) represents the invocation of a selected test operation object" (column 13, lines 27-29). As specified in claim 1, "invocation of a selected test operation," refers to representing an object with an object identifier (keyword). Playback files (object files) execute one test operation (keyword) at a time,

maintaining the prerecorded ordering, until all test operations (keywords) are read in.

b. **Sequentially reading the keywords recorded in the object file and calling functions from the function library file for executing objects corresponding to the read keywords** (column 13, lines 14-36). See figure 7, item 706. Playback files (object files) are prerecorded, containing a prerecorded sequence of test operations (keywords). Figure 7 shows one test operation (keyword) is sequentially executed at a time. In the process of executing, it is inherent that a test operation (keyword) must be read in from the file. As specified in the rejection of claim 1, an execution function (execution program) calls "the desired function corresponding to the test operation object" identified by the test operation (keyword) read in.

c. **Reading one or more successive keywords following each keyword read in the step (b) as a predetermined number of function factors needed for executing each function called in the step (b), and executing each function called in the step (b).**

"The rules interface method comprises a set of functions which determine the propriety of running a particular test operation function (function corresponding to a keyword read in), or group of test operation functions in light of the current context of the application program, or the context of the test processing" (DETX 41). Rule functions are used in conjunction with the execution function (execution program), to provide a weighted influence on a weighted execution of

test operation objects. For example, "Rule functions may be used to assure that the execute interface functions for the copy or paste test operation objects will never be called until after the test operation object execute interface for the files/open menu operation is invoked successfully" (column 21, lines 20-24). Thus if copy and paste operation (keyword) is listed before files/open menu operation (successive keyword), the files/menu operation (successive keyword) will execute before the copy and paste operation (keyword) to satisfy predetermined number of function factors.

d. **Continuing the test by returning to the step (b) if at least one keyword, which is not executed, exists in the object file, and otherwise ending the test.** See figure 7, item 704. The system checks to see if more test operations (keywords) are listed in a playback file (object file). If not, the test ends, if so, testing continues.

As per claim 3, Rodrigues discloses a computer readable medium embodying the method disclosed in claim 2 and used in a test system disclosed in claim 1. Rodrigues discloses "a computer readable storage device (computer readable medium), embodying a program of computer instructions (software test method disclosed in claim 2) executable by a computer to perform method steps for integrating testing of the program with execution of the program"(column 22, lines 52-56).

10. To expedite a complete examination of the application claim 4 rejected under 35 U.S.C. 101 (nonstatutory) above is further rejected as set forth below in anticipation of applicant amending this claim to place it within the four statutory categories of invention.

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 4-5 rejected under 35 U.S.C. 102(b) as being anticipated by Parker et al (U.S. 5,781,720).

As per claim 4, Parker et al discloses a software test system comprising:

a. **A function library file that functionizes and stores commands for executing objects of the target software as functions.**

In figure 4, a GUI interface contains a Logical Screen Element Manager (function library file), which contains entries for all of the LSEs (objects) in the system (column 9, lines 66-67) and a Physical Screen element Manager (function library file), which contains lower level functions. A test driver converts commands (keywords and factor values) into function calls to execute objects contained in LSEM. LSEM in turn calls upon lower level function calls in PSEM to process the function calls. Combined, LSEM and PSEM form function library file in this test system.

b. **An object management unit for storing keywords corresponding to respective objects of the target software and for storing factor values needed to execute the functions, wherein the keywords and the factor**

values are sequentially input in an order in which it is desired to test the target software. Parker describes in column 7, lines 62-67 the use of a database (object management unit) with his invention. "Although not required, providing a high-level interface to a database (object management unit) is often desirable. The database (object management unit) can be used to provide persistent storage between scripts; store all or part of the database of the target application for the purposes of allowing the script to run independent of the configuration of the target application; store scripts, required data files, and results, etc " (column 7, lines 62-67). "Storing scripts" in a database (object management unit) would store commands (keywords and factor values) in order in a database (object management unit), which would satisfy part b of claim 4.

c. **An execution program that sequentially reads the keywords and factor values from the object management unit, that calls the functions corresponding to the factor values to execute the objects corresponding to the keywords, and that executes the called functions using factor values** (column 15, lines 9-45).

As specified in the rejection of part b of claim 4, scripts (files containing keywords and factor values) can be stored in a database (object management unit). A script is retrieved from the database (object management unit) and then read (keywords and factor values are read sequentially). Figure 5 show the process of reading a script (a file containing keywords and factor values) sequentially. Each entry in a script is sequentially read in by a test executive. A test driver

converts a command (keyword and factor values) and sends it to the LSEM and/or PSEM (function library file). See figure 4. "To drive an LSE (execute a function corresponding to the factor values to execute the object corresponding to the keyword), the test script (a file containing keywords and factor values) 315 requests a logical operation (function), such as a pushbutton press (factor value), against an LSE (object), such as a named pushbutton (keyword). The test executive 317 resolves the LSE's logical name (keyword) into GUI-specific name and calls a function within the test driver 320. The test driver 320 then uses interface 322 to send mouse or keyboard events to the GUI 307, using the name (keyword) by which the GUI 307 recognizes the LSE (object) to designate the LSE (object) as the intended recipient for the events. The GUI 307 sends these events on to the named (keyword) LSE (object)" (column 13, lines 14-24). "The decision of which events to generate is determined by a combination of the specific function (keyword) which was called and the specific properties (factor values) of the LSE (object) which was specified as an argument to that function" (column 13, lines 33-36). The factor values in combination with a keyword, that identifies the LSE (object) as the "intended recipient for the events," are used to call events (functions) to execute LSE objects.

As per claim 5, as rejected in claim 4, Parker discloses:

a. **A user interface for displaying an input window so that the keywords and factors values are sequentially input in a testing order.**

In order to create a test script file that can be read in by a computer, the test script file has to be created on a text window and saved. Therefore it is inherent that an "input window" is available to sequentially input "keywords and factor values" in a testing order. In the present invention disclosed by Parker, manual entries can be inputted into a GUI interface in the form of keyboard or mouse click events.

b. **An object database for sequentially storing the keywords and factor values input through the user interface.** Script files created on a "user interface" as specified in the rejection of part a, of claim 5, are read into the test tool, as seen in figure 4. The test tool as specified in the rejection of claim 4 stores script files (containing keywords and factor values) in a database.

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. To expedite a complete examination of the application claim 6 rejected under 35 U.S.C. 101 (nonstatutory) above is further rejected as set forth below in anticipation of applicant amending this claim to place it within the four statutory categories of invention.

Claims 6, 7 rejected under 35 U.S.C. 103(a) as being unpatentable over Parker et al (U.S. 5,781,720) as applied to claims 4-5 above and further in view of Paley (U.S. 6,457,152).

As per claim 6, Parker et al discloses a software test system comprising of:

a. A function library file that functionizes and stores commands for executing objects of the target software as functions as specified in the rejection of claim 4.

b. A script analyzing unit that extracts keywords and factor values in an order in which the target software is tested from scripts generated when a first test is performed. The invention disclosed by Parker can create (generate) high-level script files as the output of a recording session (a first test performed) (column 25, lines 67, column 26, lines 1-3). Parker also teaches that "the test executive "takes" the commands (keywords and factor values) in a particular T script (script file) and sends them to the test driver over a communication medium" (column 7, lines 15-18). When the test executive "takes" commands (keywords and factor values) it can be inferred that it extracts commands (keywords and factor values).

b. An object management unit that stores keywords corresponding to respective objects of the target software and that stores factor values needed for executing the functions as specified in the rejection of claim 4.

c. An execution program that sequentially reads the keywords and the factor values from the object management unit, calls the functions

corresponding to the factor values for executing the objects as specified in the rejection of claim 4.

Parker does not teach **an object management unit wherein the keywords and the factor values are sequentially input after being extracted by the script analyzing unit**, but he does show GUI objects are stored in tables (object management unit) and scripts are stored in a database (object management unit). "Since all of the information needed about GUI objects is stored in tables (object management unit) within the GUI, the test driver, can either directly examine these tables in memory or make function calls to the GUI in order to retrieve information about LSEs (objects)" (column 12, lines 52-56). If the test driver (part of the execution program disclosed in the rejection of claim 4) examines the tables (object management unit) directly and holds a copy of a command (keyword and factor values) read in by the test executive, the driver can easily add the command (keyword and factor values) to the tables (object management unit), if given a reason to do so.

Paley et al teaches:

An object management unit that stores keywords corresponding to respective objects of the target software and that stores factor values needed for executing the functions, wherein the keywords and the factor values are sequentially input after being extracted by the script analyzing

unit (column 11, lines 19-20). "All of the above commands (keywords and factor values in a script file) reside in a table called the Action_list (object management unit). Within this memory table (object management unit), referred to herein as an "Action List," addresses indicate which command or transfer has been chosen" (column 5, lines 32-35). In order to store commands (keywords and factor values) from a text file (script file) it is inherent that the commands (keywords and factor values) are extracted from the file. Commands (keywords and factor values) are stored in a table (object management unit) "to allow a test to specify the order in which these commands (keywords and factor values) are run." The table can be used to look up subsequent commands to provide preconditioning information for a current command and allow for order to be changed through the protocol, rather than altering the order in a file.

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to store commands (keywords and factor values) extracted from a recorded (generated when a first test is performed) script file into a table (object management unit).

The modification would have been obvious because one of ordinary skill in the art would have been motivated to add an interactive element to the invention disclosed by Parker et al, which would allow for out-of-order reading of script files based on an intended sequence desire by a user, without rearranging the commands (keywords and factor values) in a script file. Including commands (keywords and factor values) in tables (object management unit) would be

possible because Parker already stores GUI objects in tables (object management unit), and the tables (object management unit) can be directly examined by the test driver (part of the execution program disclosed by Parker), which hold a copy of a command (keyword and factor values) used in examining the tables (object management unit).

As per claim 7, Parker et al, in view of Paley discloses a software test method comprising:

(a) **Extracting keywords corresponding to respective objects of the target software and factor values for executing the functions from a test execution script file, which is generated when the target software is executed in a predetermined testing order, and building an object database by sequentially storing the extracted keywords and factor values in a testing order,** as specified in the rejection of claim 6.

(b) **Sequentially reading the keywords and factor values from the object database and calling functions corresponding to the read keywords** as specified in the rejection of claim 6. An object management unit is an object database.

(c) **Executing the called function using the factor values read in the step (b)** as specified in the rejection of claim 6.

(d) **Continuing the test by returning to the step (b) if at least one keyword which is not executed exists in the object database, and otherwise ending the test.** As specified in claim 6, it would have been obvious to one of ordinary skill in the art to store keywords and factor values in an object database in the invention disclosed by Parker. Parker teaches in figure 5, item 400, to continue reading the next step (next keyword and factor values) from test script.

Parker does not specifically teach **ending the test** if no keywords are left. Parker teaches if an error is encountered during the reading of a test script ending the test, as see in figure 5, item 437.

Paley teaches testing continues when some entries are left to process (in the object database) and ends when "no entries left to process." It is common knowledge to one of ordinary skill in the art that ending the test when "no entries left to process" prevents an infinite loop and duplicating a test sequence.

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to end testing of a test script if no more entries are left to process.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to add end the test disclosed in Parker et al if no more steps are listed in a test script to prevent an infinite loop error in running the software test method.

13. Claims 8,9 rejected under 35 U.S.C. 103(a) as being unpatentable over Parker et al and Paley et al as applied to claims 6-7 above, and further in view of "Jargon" (Williams, Cummings 1993).

As per claim 8, as rejected in claims 6-7, Parker teaches in view of Paley:

(a1) generating the test execution script file by executing the target software in an order in which it is desired to test the target software as specified in the rejection of claim 6.

(a2) storing the keywords of the generated test execution script file in arrays having a predetermined memory space and providing an address for accessing each of the arrays as specified in the rejection of claim 6.

(a5) temporarily storing the keyword, factor values, and the address of the word corresponding to the syntax in rows as specified in the rejection of claim 7.

Parker further teaches:

(a3) sequentially searching the arrays to determine whether or not a syntax characterizing a predefined function exists, and if such a syntax does not exist, ending the test. Figure 5 shows the process of name resolution and executing commands being done sequentially. Proper name resolution of an object identifier (keyword) and its factor values require following syntax rules. The test tool uses these pieces (keyword and factor values) of information to

construct a string (array) that uniquely identifies a window in a particular GUI under test (column 20, lines 40-42). The invention disclosed by Parker requires keywords and factor values to fit an appropriate syntax. If the syntax rules are not followed, "proper [name] resolution (characterizing) of the LSE name (keyword specifying a predefined function) is not possible". The object will not be properly identified (if such a syntax does not exist) and the test will fail (ending the test) (column 22, lines 11-19).

(a4) If a word corresponding to the syntax in the step (a3) exists, extracting the factor values located within a predetermined distance from the word by searching arrays in front of and behind the word and providing a keyword needed for calling the corresponding function. Parker discloses the use of a parser to parse a command line that divides and takes (extracts) a command line. The parser searches a command line and identifies window tags (factor values), window names (keywords), window ids (keywords), and class names (keywords) by searching for specific identifiers. In this case the parser searches a string (array) until it reaches a specific identifier (dot operator, a backslash).

Parker does not specifically teach **sorting in rows the keywords, factor values, and the address of the syntax stored in the step (a5) according to the address of the syntax** and **storing the keywords and factor values in rows in the object database in the order as sorted in the step (a5).** Parker in view of Paley teaches an object database that stores keywords and

corresponding factor values as specified in the rejection of claims 6 and 7. Parker teaches a database disclosed in the present invention "can be used to provide persistent storage between scripts; store all or part of the database of the target application for the purposes of allowing the script to run independent of the configuration of the target application; store scripts, required data files, and results, etc " (column 7, lines 62-67).

It is known to one of ordinary skill in the art that steps (a6) and (a7) specify a database engine as disclosed by Jargon (Williams, Cummings 138). Database engines are added to a database to aid in sorting a database that contains different types of information.

It would have been obvious to one of ordinary skill in the art to add a database engine to the database disclosed in Parker et al.

The modification would have been obvious because database engines are used to sort a database that contains more than one field type. Parker discloses a database that contains script files, and other information to aid in testing, therefore it would have been obvious to add a sorting function (database engine) to sort the database, keeping script files, keywords and factor values, and results separate from each other in a predetermined memory address for quick of information, without traversing the entire database.

As per claim 9, Parker in view of Paley teaches a software test program that runs in the software test system defined in claim 7 and comprises of the method

disclosed in claim 8. Parker teaches a software test method embodied on a computer readable recording medium. The test method disclosed by Parker runs on an operating system with a GUI interface. The invention disclosed by Parker is used on "cross-platform application development." "Cross-platform means pertaining to more than one GUI.

Paley teaches, "functionality (software test program) may be embodied in computer readable media (or computer program product) such as 3.5 inch diskettes (recording medium) to be used in programming an information-processing apparatus to perform in accordance with the invention" (column 4, lines 16-22).

It would have been obvious to a person of ordinary skill in the art at the of the invention was made to embody the software test program disclosed by Parker in view of Paley on a computer readable recording medium.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to store the software test program disclosed by Parker in view of Paley on a computer readable recording medium to perform software tests on more than one GUI interface.

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Duggan et al. US 6,002871 "Multi-User Application Program Testing Tool"
- Thai, Victor M. US 6,047,389 "Testing of a Software Application residing on a Hardware Component"
- McKeeman et al. US 5,905,856 "Method and Apparatus for Producing a Software Test System using Complementary Code to Resolve External Dependences"
- Beers et al. US 6,279,124 "Debugging Tool for Linguistic Applications"
- Ottensooser, Avner US 5,574,855 "Determination of Software Functionality"
- Brouwer et al. US 6,002,869 "Method and System for Testing Hardware and/or Software Applications"
- Rosich et al. US ,574,855 "Method and apparatus for Testing RAID Systems"

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Noreen Moiduddin whose telephone number is (703)305-0358.

The examiner can normally be reached on Monday, Tuesday, Wednesday, Thursday, and Friday from 9:00 AM to 5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisory Kakali Chaki can be reached at (703)305-9662. The fax number for this group is (703)308-3988.

JOHN CHAVIS
PATENT EXAMINER
ART UNIT 2124